MA 302 – Spring 2019
Mid-semester project (due Tuesday, March 19)

Instructions:
The code submitted should be your own creation. You may consult MATLAB's documentation or the notes from class. The submission of codes obtained from online sources is a violation of Loyola's honor code and may result in failure of the course.

Problem 1: Euler's method

In class we have discussed Euler's method for solving initial value problems of the form

$$y'(t) = f(t, y(t)) \quad a \le t \le b$$
$$y(t_0) = y_0$$

The method proceeds as follows:

1. Divide $[a, b]$ into N subintervals of size $h = \frac{b-a}{N}$ to obtain
$$a = t_0 < t_1 \cdots t_{N-1} < t_N = b$$
2. Starting with y_0, compute the solution at t_k for $k = 1, 2, \cdots, N$ as
$$y_{k+1} = y_k + hf(t_k, y_k)$$

Write a function `[t_vals, y_vals] = approx_ode(f,a,b,inital_value,n)` that takes as input a function defined in a file `yprime = f(t,y)`, defined in `f.m` and two values a and b indicating the left and right intervals on which the initial value problem is defined and `n`, the number of Euler steps. Your function should return 2 vectors (`t_vals` contains the values $t_0, t_1, \cdots, t_N$ and `y_vals` is the solution at those points $y_0, y_1, \cdots, y_N$) Test your code on the initial value problem

$$\frac{dy}{dt} = 1 + \frac{y}{t} \quad 1 \le t \le 6, \quad y(1) = 1$$

whose exact solution is $y(t) = t(1 + \ln t)$. Run your code for $h = 0.25$ and plot the solution. Your plot

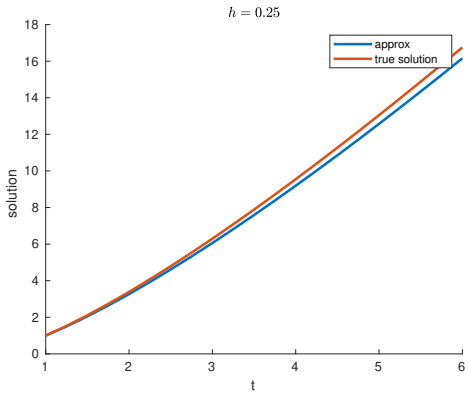

Figure 1: Numerical solution of IVP

should look like Figure 1. Finally, confirm that the error at $t = 6$ is 0.5948.

Submit the following:

- All function and script `m`-files.
- The figure `prob1.jpeg`
- A summary of your implementation and a comment on why the approximate solution appears to slowly move away from the true solution as t increases

Problem 2: Newton's method on the complex plane

In class we discussed **Newton's method** for approximating nonlinear equations of the form $f(z) = 0$. Starting with an initial guess z_0, the method generates a sequence $\{z_n\}$ which usually converges to the root. The method is

$$z_{k+1} = z_k - \frac{f(z_k)}{f'(z_k)}$$

In this problem you will visualize the performance of Newtons method on the **complex plane**. Consider the function $f(z) = z^3 - 1$. It has three roots, one is real and the other two are complex:

$$r_1 = 1, \quad r_2 = -\frac{1}{2} + \frac{\sqrt{3}}{2}\boldsymbol{i}, \quad r_3 = -\frac{1}{2} - \frac{\sqrt{3}}{2}\boldsymbol{i}$$

Using the Newton's iteration

$$z_{k+1} = z_k - \frac{z_k^3 - 1}{3z_k^2}$$

and **complex arithmetic**, the sequence $\{z_k\}$ will converge to one of the 3 roots depending on the initial guess z_0. In this problem you will write a script to visualize which starting point leads to which of the 3 roots. The set of initial values that lead to convergence of the same root is called the *basin of attraction* of that root. Your script is going to produce the *basin of attraction* for this problem.

1. Create a grid of points on the complex plane $\{\boldsymbol{z}_0 = x + iy | -1 \leq x, y \leq 1\}$ with 1000 points in each direction. Each point on this grid will be an initial guess.
2. Perform the Newton iteration
$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k - \frac{\boldsymbol{z}_k^3 - 1}{3\boldsymbol{z}_k^2}$$
for all points in your grid at once for $k = 1, \cdots, 50$. Here we pick 50 to ensure that every point has converged to the root.
3. Assuming that `Z`$= \boldsymbol{z}_{50}$ is the set of values from your Newton loop, you can call
```
>>unique(round(imag(Z)))
```
to confirm that your initial values have converged to 3 different roots. Explain how the output of the following command shows that all the points have converged to 3 roots.
4. Use the following commands to visualize your basin of attraction
```
>>image((round(imag(Z))+2)*20);
```
Save your image in a file `prob2.jpeg` (your image should look like Figure 2)
5. Write commands to determine the number of points on the grid that have converged to each root.

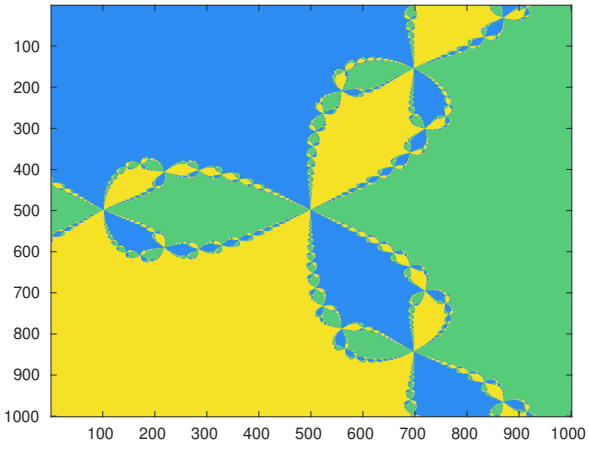

Figure 2: Attraction basin for $f(z) = z^3 - 1$

Submit the following:

1. All function and script `m` files.
2. Solutions to questions asked above
3. The image `prob2.jpeg`

Problem 3: More chaos

Generalize the `chaos.m` script to a 5 sided die and a regular pentagon with coordinates

$$
\begin{aligned}
&0 + \boldsymbol{i} \\
&-\frac{1}{4}\sqrt{10 + 2\sqrt{5}} + \frac{1}{4}(\sqrt{5} - 1)\boldsymbol{i} \\
&-\frac{1}{4}\sqrt{10 - 2\sqrt{5}} - \frac{1}{4}(\sqrt{5} + 1)\boldsymbol{i} \\
&\frac{1}{4}\sqrt{10 - 2\sqrt{5}} - \frac{1}{4}(\sqrt{5} + 1)\boldsymbol{i} \\
&\frac{1}{4}\sqrt{10 + 2\sqrt{5}} + \frac{1}{4}(\sqrt{5} - 1)\boldsymbol{i}
\end{aligned}
$$

Submit the following:

1. A summary of modifications you made in chaos.m
2. Your modified `chaos.m` file
3. Your final image after a run with $N = 5000$ points `prob3.jpeg`
4. Rerun your code with new points generated with a cut factor of 0.618 save that in `prob3b.jpeg`

Submission of exercises

Place all your files (`m-files, summary.txt, diary.txt`) in a folder named `lastname_project` and zip the folder to create a file `lastname_project.zip`. Email your zip file `lastname_project.zip` to `pchidyagwai@loyola.e` with subject `MA302_project`.